A DYNAMIC MODEL FOR COMPUTER-AIDED CHOREOGRAPHY

Carol Withrow

Utah University

Prepared for:

Advanced Research Projects Agency
Rome Air Development Center

June 1970

A DYNAMIC MODEL FOR COMPUTER-AIDED CHOREOGRAPHY

by

Carol Withrow

June 1970

UTEC-CSc-70-103

This research was supported in part by the University of Utah Computer Science Division and the Advanced Research Projects Agency of the Department of Defense, monitored by Rome Air Development Center, Griffiss Air Force Base, New York 13440, under contract AF30(602)-4277. ARPA Order No. 829.

1. INTRODUCTION

The application of the digital computer in the field of dance choreography is a relatively unexplored area. The use of the computer in other arts is more widely known. The computer has, for example, been used to compose, arrange and analyze music, as well as to produce musical sounds. It has generated graphical works of art. Without entering the controversy over the computer's "true creativity," one can state that the computer has had some impact on the arts and the imagination of artists. Its future impact will no doubt be greater as its capabilities become explored and understood.

In the field of choreography the influence of the computer has so far been negligible. To understand the possibility of computer applications in this field, one should first consider the way in which a choreographer works.

His modus operandi is typically this: He assembles a company of dancers, and then, bringing with him only a general idea of what he wants, he begins experimenting and creating his work of art. The dancers are his living tools. Upon completion, the record of the evanescent creation that is a dance lies only in the minds of choreographer and dancers and, perhaps, on film.

The accumulation of a choreographic literature has been hampered primarily by the existence of two interrelated problems. These are the lack of an adequate movement

notation and the enormous amount of information inherent in the description of even a simple human movement.

Dance notation systems do exist, the most successful probably being Labanotation, devised by Rudolf Laban and first published in 1928 [1]. In this system, positions of various portions of the body are written on a staff resembling a musical staff. The staff lines are vertical and are read from bottom to top. An example of this notation may be seen in Appendix A.

Transcription of dances into this type of written notation is time-consuming and difficult, and is not a frequent practice of choreographers.

Both of the above mentioned problems (the lack of a good notation system and the large information content of movement) concern the manipulation of information. Clearly here is unbroken ground for the information scientist.

Thus far, the only significant use of the computer that has been made in the field of dance has been to use it to produce lists of instructions for the dancer [2]. Using a set of symbols representing possible movements, a computer program generates a list of movements to be performed in the given order. An element of randomness is incorporated into the algorithm for producing these instructions, in an effort to break away from established patterns of movement. Such instructions are quite general because of the two problems discussed above, thus leaving a large amount of interpretation

to the dancer.

Noll [4] has suggested that a different approach to choreography be tried, making use of the interactive graphical display systems which are now coming into use:

> Instead of using the dancers as his choreographic instrument, the choreographer interacts with the computer during the creative process. Stick figure representations of the dancers appear in some form of three-dimensional display on the face of an electronic display tube. The choreographer, by manipulating different buttons on the console, controls the movement and progress of the work: Different movements might be stored in the computer's memory and put together at will. Individual movement restrictions for each dancer could even be introduced into the process. Various elements of chance and randomness could be used at the discretion of the choreographer. Scenery and stage props could also be specified by the choreographer and drawn by the computer. In effect, all the different aspects of the work would be under the direct control of the choreographer to manipulate and experiment in a myriad of combinations. When desired, certain portions of the creative process could be filled in by the computer which might be programmed to learn the choreographer's style. All this is a completely new creative process, and might result in new dance forms.

Such a tool would be a solution to the dance notation problem. It would also be useful for pedagogical purposes, as a laboratory tool for student choreographers, and as a device for producing permanent records of dances. Though not every choreographer would wish to use the system described, it would provide a means of research into human movement, choreographic styles and the choreographic process itself.

It is with such a program in mind that the present work was undertaken. The prime problem upon which attention is focused is this: How is a mathematical model of the

human body to be manipulated without resort to complex mathematical formulas? How can a dancer model be controlled or driven in a manner sufficiently simple that it not interfere with the creative process of the artist?

Noll suggested manipulating buttons on the console. Another approach that has been tried is an anthropometric harness [6]. This device is attached to a person, and the movements of various key positions on his body are recorded. These movements are then shown on an electronic display as a stick figure going through the same motions as the harness-wearer. Possibly the interactive device known as a joystick, a movable shaft whose position can be sampled, would be useful in the solution of the problem.

This dissertation describes an interactive program which relates the angular movements of the joints of the display model to hand-drawn curves. These curves are drawn with a stylus on a graphics tablet. The pen and tablet are part of an interactive system that includes a cathode ray tube display, a digital computer, and the user himself. This program demonstrates that this method of model-manipulation is feasible.

This dissertation thus demonstrates that the notation problem can be circumvented by the implementation of this type of graphics program as envisioned by Noll. The vast capacity of the computer to store and manipulate large amounts of complex information is here brought to the problem

of representing human movement. The solution to this problem is thus initiated, and it is hoped that the effort described here might be of value to future workers.

II. PROTOTYPE PROGRAM

II.1 Description

This program is basically the type of interactive program described in the introduction; that is, a user sits at a display console and views a dynamic display. The picture in this prototype program is a stick figure with two moving joints. The information that causes the console to display the figure in its various positions is generated in the computer and is based on user-supplied inputs. These inputs are primarily curves drawn on a tablet with a stylus. These curves are shown on the display. Using the coordinates of successive points along the curves, the computer then calculates arrays of angles through which the moving parts of the stick figure will move. The illusion of movement is then created as the computer displays in rapid succession these various positions.

There are four different basic displays that the user sees. These will be referred to as "pages" in the remainder of this discussion.

The initial display is page one, the table of contents, which merely lists the three pages that follow it (Fig. II.1-1). Pointing to the desired page number or name with the pen causes that display to appear, as if pages had been turned.

Page two (Fig. II.1-2) is concerned with determining the movement of the model's right leg about a hip joint. The user sees a small fixed figure with the pertinent joint indicated and three rectangles in which the curves are to be drawn. These curves will represent the angular displacement of the leg about the X, Y, and Z axes. Also displayed is a small diagram of these axes, showing their positive directions. The remainder of the display consists of the page number and a list of user options. The options, which are chosen by pointing with the pen, include drawing, saving or erasing curves; returning to the table of contents; and displaying previously drawn curves. Turning pages of the display can be accomplished by pointing to the arrows at the top of the picture by the page number. The right arrow indicates advancing one page; the left arrow indicates turning back one page.

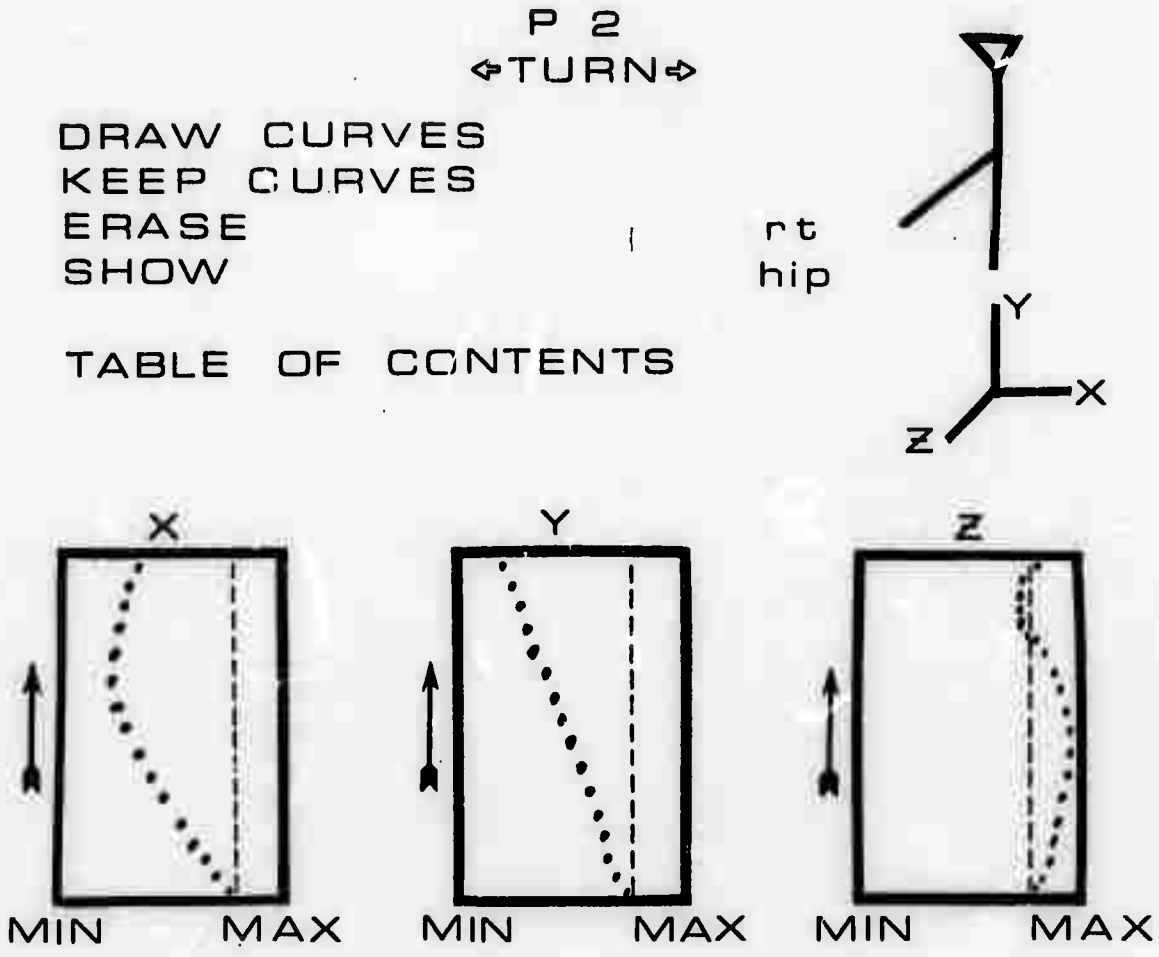


FIGURE II.1-1

FIRST DISPLAY

P 1
TURN⇨

A DYNAMIC BODY MODEL

TABLE OF CONTENTS

FIGURE II.1-2

SECOND DISPLAY

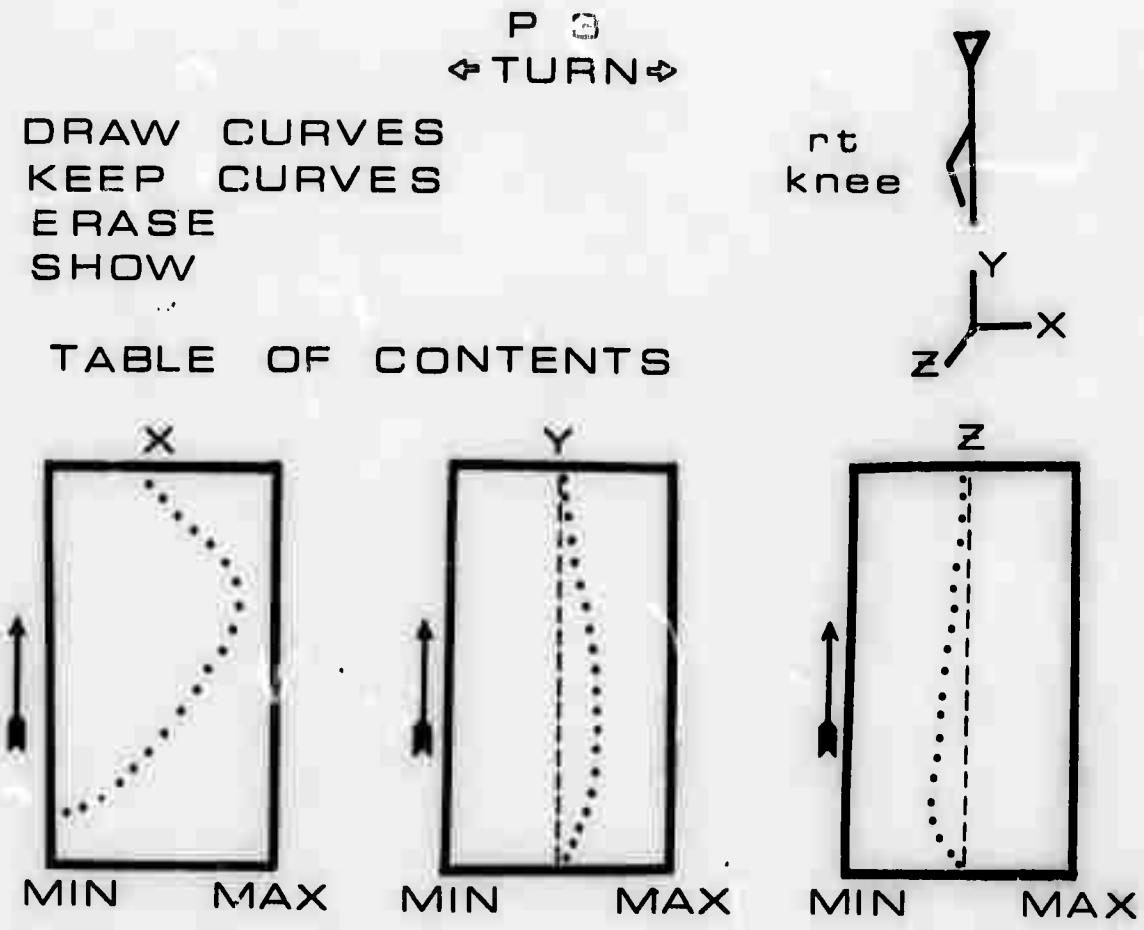


FIGURE II.1-3

THIRD DISPLAY

P 4
⇦TURN

SHOW MOVE
ROTATE X
ROTATE Y
STOP

TABLE OF CONTENTS

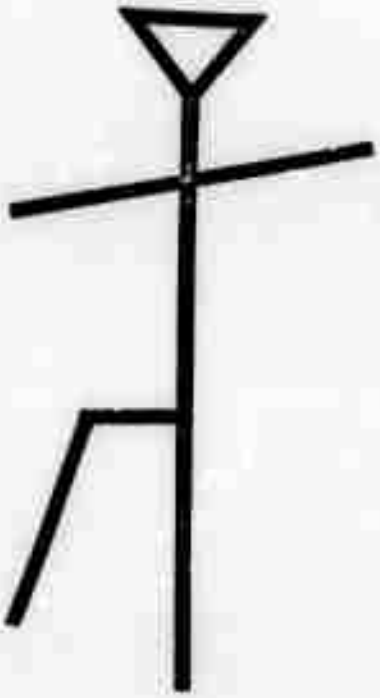
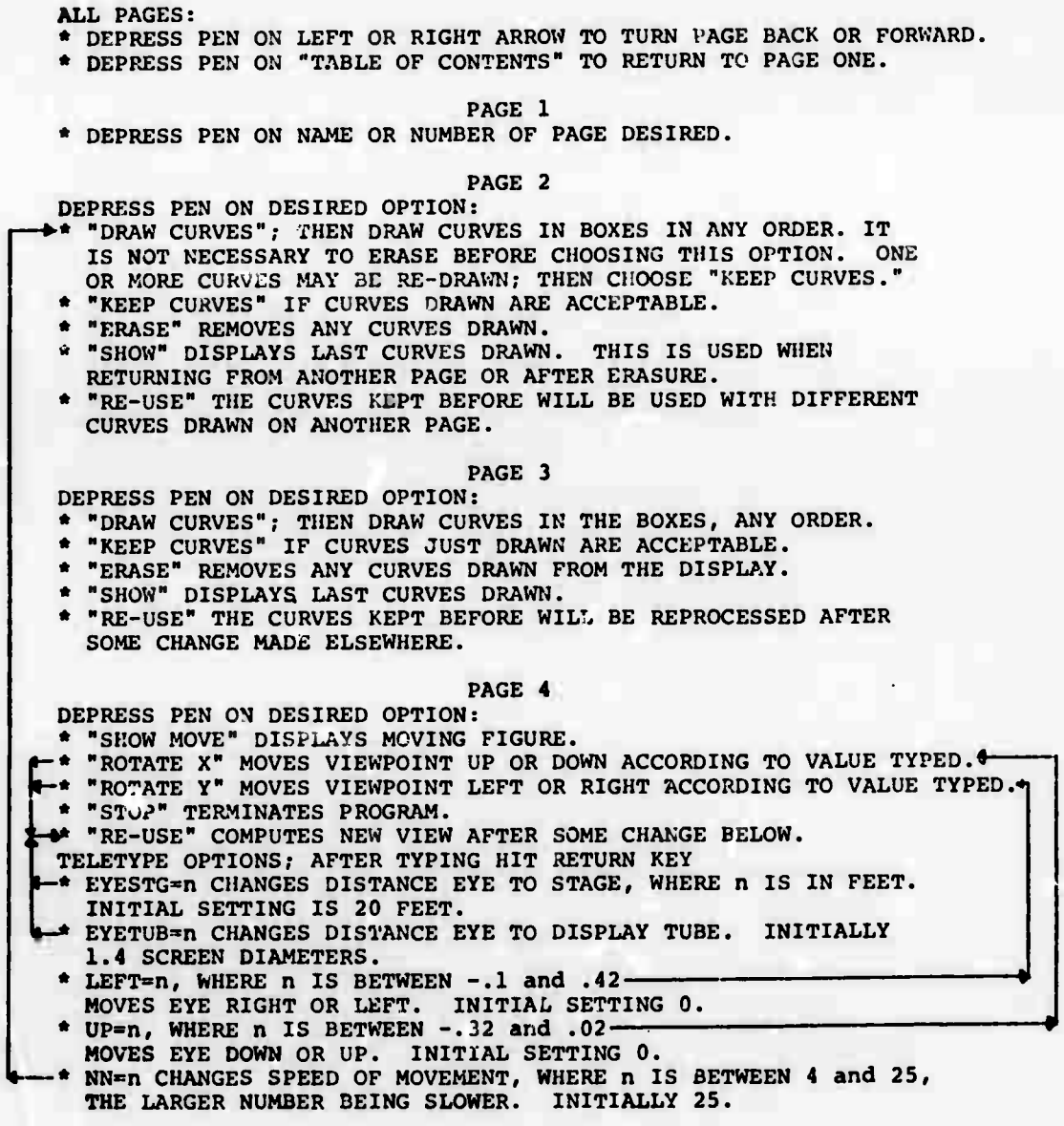


FIGURE II.1-4

FOURTH DISPLAY

DYNAMIC MODEL PROGRAM

INSTRUCTIONS FOR THE USER



USER'S MANUAL

FIGURE II.1-5

Page three (Fig. II.1-3) is similar to page two, but is concerned with the angle at the knee. As movable joints are added to the program, similar pages may be added. The program is so structured that it may be extended in this way. This will be discussed further in Section III.

Page four (Fig. II.1-4) displays a larger stick figure and a set of user options. If the option to display the current movement is selected, the stick figure is shown in varying positions in rapid succession, so that he appears to move. This dynamic display may, of course, be repeated as many times as desired. Page four has the usual options for changing the display to other pages.

Also associated with page four are options to change the positions of the viewpoint and the stage, and to alter the speed of the figure's movement. These will be explained more fully in Sections II.2-4 and II.4.

As he sits at the console, the user has beside him a one-page manual of instructions for using the program. This manual is intended for use by non-technical persons with no programming experience, and is shown in Figure II.1-5.

II.2 Data Definition and Manipulation

All objects in this program are ultimately described in a three-dimensional coordinate system having its origin at the viewpoint. The stage and each joint of the stick figure have associated local coordinate systems. For the hip joint, this local system originates at the base of the

spine, and the vertical (Y) axis corresponds to the spine. For the knee joint, the system has its origin at the knee point, and the vertical axis corresponds to the upper leg vector, whatever orientation that vector may have.

In addition, each joint has associated with it the following:

1 initial point coordinate vector

3 x n array of angles

6 angle constraints

3 x 3 natural linkage rotation matrix

4 x 4 update transformation matrix

3 table pointers

4 x n display point array

where n is the number of points in time for which sets of display points have been computed. These will be explained in the sections that follow.

II.2-1 Angle Determination

Each moving joint has associated with it an array of angle values, representing the different positions of the limb or vector at different points in time. These angle values are relative to the local coordinate system. Each angle is divided into three components representing the angular displacements about the three axes. Each of these three components may be thought of as an array corresponding to one of the user-drawn input curves.

The curve-drawing rectangles are labeled "X," "Y," and

"Z" for the axes about which the angular displacements are measured. The vertical arrows seen on the display indicate that the positive direction of the independent variable, time, is from the bottom to the top of the rectangle. This orientation was chosen to correspond to that of Labanotation, which is also read from the bottom upwards. Within each rectangle is a vertical dashed line, indicating the reference angle of zero degrees of rotation about that axis. The left and right sides of the rectangle, labeled minimum and maximum, represent angles of 100% of the constraint angles.

A constraint angle is set by human limitations and is the limiting angular displacement of a limb about an axis. In this program each angular component has negative and positive constraints. The left (user's left) side of the displayed rectangle represents the constraint when the limb is rotated in a negative direction about that axis; the right side, positive. Some joints, such as the knee, will have at least one constraint of zero degrees.

Thus it is the curve's amplitude, or horizontal distance of the curve from the dashed line, that determines the angular component at each time point. This component represents the total angular displacement about one axis at one time point and is made to be directly proportional to the input curve amplitude.

The angular component is computed from this formula:

$$\Theta = (a/d)K$$

where Θ is the angular component, a is the amplitude of the

input curve (in display scope units), d is the distance between the dotted line and box edge (in scope units), and K is the constraint angle (in radians) in either positive or negative rotation, depending on which side of the dashed line the curve point is.

The angle constraints are constants entered in just one place in the program, in the data section, and are easily altered.

The dashed line is located to reflect accurately the ratio between the absolute values of negative and positive constraints. If constraints are changed, the dashed line is automatically repositioned.

It is seen that this method of angle determination obviates the need for checking angles computed against limiting constraints. It is unnecessary to guard against computing unrealistic positions, because it is impossible to exceed the constraints. Constraints will be discussed further in section III.

II.2-2 Vector Determination

Homogeneous coordinate representation is used to represent point coordinate vectors and transformation matrices [7] [8]. Homogeneous representation is simply the representation of an n-dimensional object in n+1 coordinates. For example, a point coordinate in three-dimensional space is represented by four coordinates, the fourth one being the homogeneous coordinate.

The reason for using homogeneous coordinates is that all rotations and translations of a particular vector can be represented in a single matrix.

Each limb is represented initially as a row vector in local coordinates, and is oriented correctly by multiplying it by its updated transformation matrix, which contains all the rotations and translations through which it has passed, including that of previous systems to which it is related.

The transformation matrix can be viewed as consisting of components:

$$\begin{bmatrix} \begin{bmatrix} 3 \times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} X & Y & Z \end{matrix} & 1 \end{bmatrix}$$

The upper left 3x3 matrix represents the rotation, while the first three values in the bottom row represent the translation in the X, Y, and Z directions, respectively. (The last column represents perspective transformations but is not used as such in this work [10].)

The rotation matrices used were

about the X axis:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & C & S \\ 0 & -S & C \end{bmatrix}$$

about the Y axis:
$$\begin{bmatrix} C & 0 & -S \\ 0 & 1 & 0 \\ S & 0 & C \end{bmatrix}$$

about the Z axis:

$$\begin{bmatrix} C & S & 0 \\ -S & C & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where C = cos Θ, S = sin Θ. Θ = angular displacement about one axis.

An example of a unit vector on the X axis being rotated about the Y axis follows.

$$[1 \quad 0 \quad 0 \quad 1] \begin{bmatrix} C & S & 0 & 0 \\ -S & C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [C \quad S \quad 0 \quad 1]$$

Translating this resultant vector a units along the X axis, b along the Y and d along Z would be performed as follows.

$$[C \quad S \quad 0 \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & d & 1 \end{bmatrix} = [C+a, S+b, d, 1]$$

Both operations could be performed by one multiplication, where the transformation matrix, M, is the product of R, the 4x4 rotation matrix (in homogeneous representation), and T, the translation matrix:

M := R x T

For the above example,

$$M = \begin{bmatrix} C & S & 0 & 0 \\ -S & C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & d & 1 \end{bmatrix}$$

and

$$[i \quad 0 \quad 0 \quad 1] \begin{bmatrix} C & S & 0 & 0 \\ -S & C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & d & 1 \end{bmatrix} = [C+a, S+b, d, 1]$$

as before.

Note that this represents the rotation first followed by the translation. The reverse order would give a different result.

The rotation matrix is the product of the natural linkage rotation matrix, the three rotational components of the local system, and the rotation matrix of the previous or "father" system to which the local system is linked. For example, if the local system is that one with its origin at the knee, the rotation matrix of the hip joint must be included in the product that forms the knee updated rotation matrix. In turn, the rotation matrix of the hip joint includes rotations of the spine relative to the stage, and of the stage relative to the viewpoint.

The natural linkage rotation matrix describes the orientation of the local coordinate system initially (at rest) to the system just preceding it. These are all identity matrices in this program, but they are included for generality.

The translational portion of the transformation matrix represents the origin of the local coordinate system.

Once the transformation matrix is set up, the position of the vector it represents is determined by the following multiplication,

$$\vec{V}_k = \vec{V}_o \times [M]$$

where $\vec{V}_o$ is the initial point coordinate row vector, M is a 4x4 transformation matrix, and $\vec{V}_k$ is the translated and rotated point coordinate vector. $\vec{V}_o$ is represented in local coordinates; for example,

$$\vec{V}_o = [100, \quad 0, \quad -5, \quad 1]$$

would represent a point that was translated 100 scope units from the origin along the X axis, zero along the Y axis, and minus five along the Z axis of its associated local coordinate system. The homogeneous coordinate in its most general case is a scale factor by which the entire vector is divided, but in this work it will always have a value of one.

II.2-3 Perspective Transformation

The points obtained by the process just described have

coordinates based on a system with its origin at the eye or viewpoint. These are converted to display points for a perspective picture [11] with the following formulas:

$$X_d = (X_j Z_s)/Z_s - X_s$$
$$Y_d = (Y_j Z_s)/Z_j - Y_s$$

where (X_d, Y_d) is the displayed point in screen coordinates, (X_j, Y_j, Z_j) is the object point in eye coordinates, and (X_s, Y_s, Z_s) is the screen origin in eye coordinates (See Fig. II.2.3-1).

The values of Z_s and Z_j (the distances from viewpoint to screen and object) may be varied while using the program, as will be explained in the next section. The user may thus obtain the perspective from the distance desired, and experiment with subjective effects obtained by varying the viewing distance.

II.2-4 Camera Positioning

The ultimate origin of all points in all coordinate systems is the viewpoint. The viewpoint, therefore, cannot be altered. The stage, however, may be repositioned relative to the viewpoint, which creates the illusion of moving the camera or viewpoint.

There are four parameters which may be changed to vary the positions of the stage and of the display scope relative to the viewpoint. These must be typed in on the Teletype while page four is being displayed. Precise instructions for doing this are to be found in the user's manual.

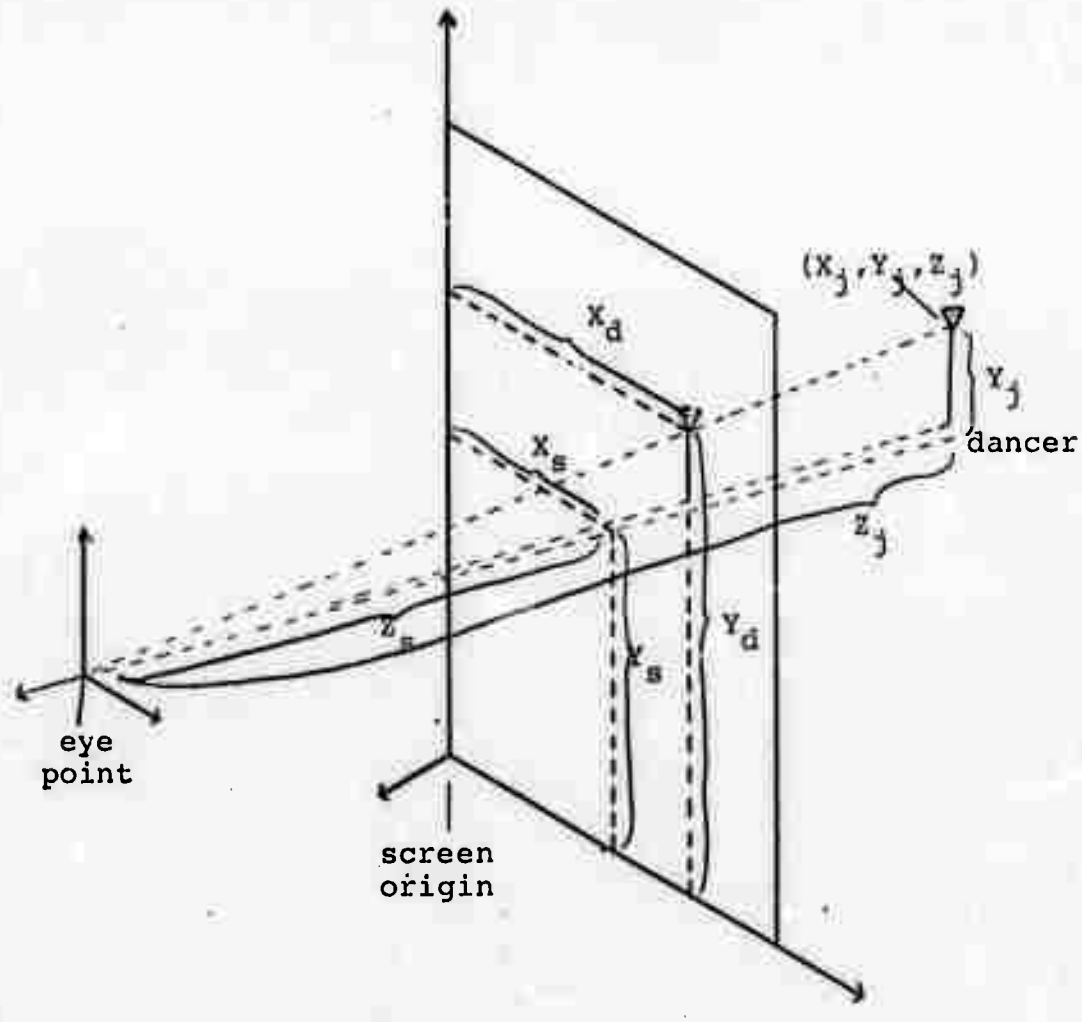


FIGURE II.2.3-1

ELEMENTS OF THE PERSPECTIVE TRANSFORMATION

The first of these parameters is the distance from eye to stage. It is represented in feet, and is initially set to twenty feet.

The distance from eye to display tube may likewise be altered by typing in a new value. This value is initially set at 1.4 x width of display tube.

The stage may be rotated about the X or Y axes of the eye's coordinate system. These rotation angles are in radians and are initially set to zero. Suggested limits within which to contain the rotation angles are included in the user's manual. Since they are represented in the same form as limb rotation angles, they are actually angle arrays. The data structure thus allows for panorama viewing, though this program implements only a static viewpoint.

II.3 Data Structure

Each of the origins of the various coordinate systems is associated with a node. The nodes are related to each other in a tree structure (Fig. II.3-1) and are triply linked by father-brother-son pointers as shown in Table II.3-1. Each node except the first points to a father node, to one "younger" brother and to a first son. The pointer value of zero in the father column indicates that that node is the origin of the tree. Pointer values of " - " in brother and son columns indicate that that relationship is unfilled, and they are actually represented in the program by values of zero.

TREE STRUCTURE WITH NODE NUMBERS

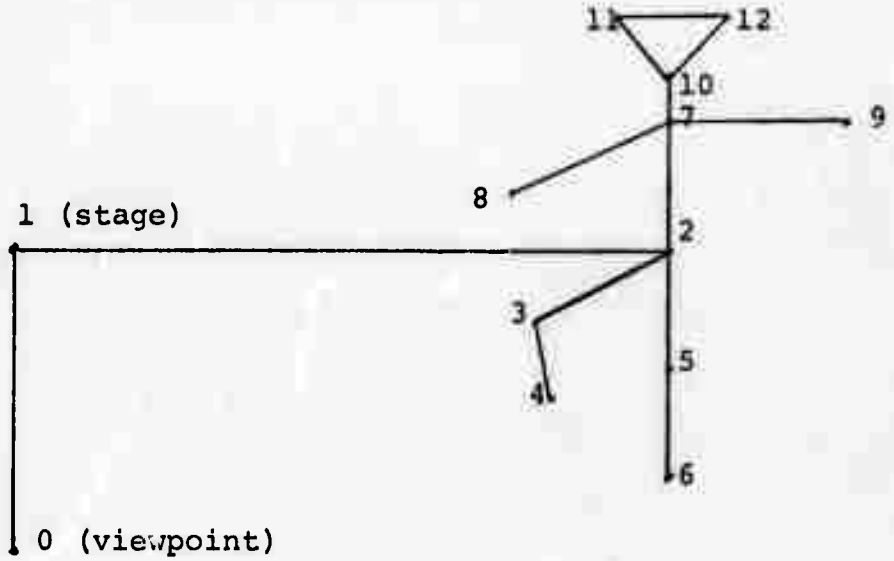


Fig. II.3-1

TABLE OF TREE RELATIONSHIPS

Node	Pointer Values		
	Father	Brother	Son
1	0	--	2
2	1	--	3
3	2	5	4
4	3	--	--
5	2	7	6
6	5	--	--
7	2	--	8
8	7	9	--
9	7	10	--
10	7	--	11
11	10	12	--
12	10	--	--

Table II.3-1

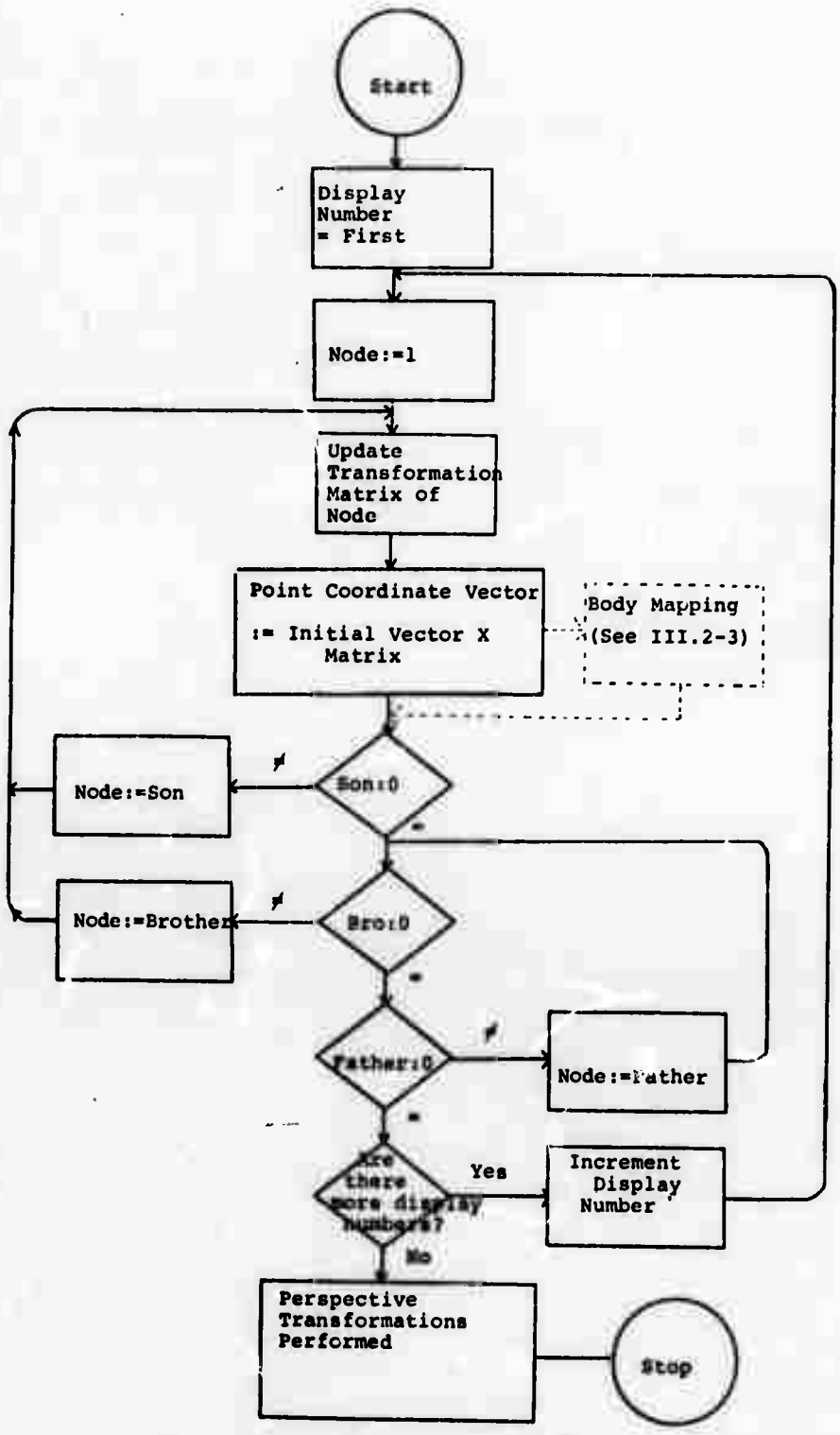


FIGURE II.3-2

ALGORITHM FOR DETERMINING COORDINATES OF DISPLAY POINTS

The algorithms for determining the display points and for displaying them are both driven by this table. The point-determination algorithm is shown in Figure II.3-2, and the display algorithm is similar. A more complex display could be handled by these key routines by entering the appropriate table and initial data in the data section of the program.

II.4 Curve-Processing

When a curve is drawn on the tablet with the stylus, the information is stored as an array of three-dimensional (X, Y, Z) points. The value of the Z coordinate is information about the continuity (beginning and ending) of the curve and is discarded after use. The X and Y coordinates are the usual horizontal and vertical coordinates (in scope units).

Once a point is recognized, a succeeding point is not recognized until the stylus has moved outside of a 4 x 4 square area centered at the preceding point.

It will be seen, then, that the number of points recognized on one of the user-drawn curves will vary from curve to curve. A straight line will consist of fewer points than a curve with many inflections. The problem thus arises of relating the curve points to each other.

It is desired that all curves contain the same number of points and, further, that points with like vertical coordinates will correspond to each other. This is accomplished

by thinning the points to a constant number of points which are evenly spaced along the vertical axis. Points are then filled in between these thinned points in the following manner: A parabolic approximation to the first four points is obtained by the least-squares method [12], [13]. Five points are then interpolated in the middle interval, and the procedure is repeated for all succeeding intervals, save the last. Interpolation is linear in the end intervals.

The number of points to which the curve is thinned is set to twenty-five in the program, but can be changed by typing in different values on the Teletype. (Instructions for so doing are in the user's manual). Changing this parameter alters the number of "frames" shown when the movement is displayed. This parameter is, then, in effect a scaling factor which varies the speed of the motion.

III. AUGMENTING THE PROTOTYPE

The prototype program is intended as a suggested approach to the solution of the problem of manipulating the figures in a choreography program. There are many factors to be considered in further development of this idea. Some of these factors will be discussed in this section.

III.1 Improving the Existing Model

There are a number of significant improvements that could be made within the context of the existing 12-node data structure and the stick figure representation of the model (See Figure II.3-1).

Full articulation of the existing model is desirable, and the existing structure is such that achieving this is primarily a clerical effort on the part of the programmer. Two things must be added to the program for each additional joint articulated: angle constraints and a new display (curve-drawing page), analogous in every way to the second display (page two). The angle constraints are entered in the Block Data section of the program. All necessary arrays are already in existence and correctly dimensioned for full articulation.

Implicit in full articulation is the movement of the dancer relative to the stage. It will be recalled that

there is a separate stage coordinate system which is between the eye (viewpoint) coordinate system and the initial or base-of-spine coordinate system of the dancer. Therefore, movement of the dancer across the stage is possible within the data structure, though it is not implemented in the prototype.

Such movement would be internally represented in transformation matrices, as are all other movements. In addition, a special array of vectors would be needed to represent the initial point coordinate vector, since this is not of constant magnitude. The problem arises of entering the latter type of information in the program. How does the user indicate the desired movement of the whole dancer relative to the stage?

Copeland [14] has written a graphics program which traces the path of the stylus on a tablet area representing the stage in plan view; one or more small triangles, representing dancers, then traverse this path. Such a program could be used to put in information about the X and Z translation (upstage/downstage, right/left), and the Y translation (vertical movement) could be found in a similar manner. A hand-drawn curve on the tablet could be interpreted as a simple linear relationship between time and the distance of the base of the spine from the floor. Obviously, rather severe constraints would have to be built in here. Perhaps the program could check the feasibility of these values for vertical movement by comparing them with the corresponding

leg positions. Eventually, such factors as momentum, gravity, and muscular strength might be included in this feasibility check.

Storage requirements of an expanded program will need to be watched. The prototype program requires a little less than 21,000 words of storage. This includes the expanded dimensions mentioned before. The Univac 1108 has a core storage capacity of 65,000 words of which approximately 53,000 are available to the user. Each new display (for each new joint articulated) adds about 3,700 words to the storage requirement, about half of which is the new page display for curve-drawing, and half of which is code. Thus, any eight of the existing unarticulated nodes could be articulated without requiring additional storage, but articulation of all twelve nodes would not be possible without some space-saving changes.

Some storage economy could be effected by packing the object point coordinates two or three to a (36-bit) word. This is possible, because these coordinates are represented internally as signed integers with known ranges. A limit would need to be placed on the Z-coordinate, the distance from viewpoint to figure. (It will be recalled that the user may vary this distance.) If it were limited to fifty-eight feet, the coordinates could be packed three to a word. Packing two to a word would permit a limit of 3745 feet.

Another improvement that could be made in the existing

program is to make the input language entirely stylus-and-tablet oriented. The need for this becomes obvious to a user attempting to exercise the Teletype-input options, such as by varying the distance of the stage. Though the Teletype is located immediately next to the tablet and the user, it is disconcerting to alternate between input devices. Programming this change would not be difficult.

Another change concerns matrix multiplication. When a rotation matrix undergoes a number of computer multiplications, as it does in this program, the orthogonality properties are lost because of computer truncation. Mahl [15] has devised a correction equation which restores orthogonality to the rotation matrix. It is a rapidly converging, iterative scheme based on a Taylor's series. As the number of matrix multiplications is increased in expanding the model, incorporation of this scheme into the program might be considered.

III.2 Anthropometric Considerations

A truer representation of the mechanical relationships of the human body, or in other words, a better stick figure, should be an early goal of one improving the prototype.

III.2-1 Constraints on Movement of Joints

In the prototype program, crude approximations of the natural constraints on joint movements are used, the model being a crude approximation of a human figure.

These limits on joint movements were assumed to be constant, although in reality most constraints will vary

according to the position of other parts of the body. The constraints will of course also vary from individual to individual, and even from day to day within the same individual [17]. However, the working assumption of constant constraints is probably no grave drawback even in a highly developed program if the constraints supplied are quite liberal. The end product of the program is a person's subjective impression of a graphical display. Unsatisfactory postures can, hopefully, be altered by the user, until they appear to be satisfactory from a variety of vantage points.

These angular constraints for joint movement have been extensively measured for medical purposes and for use in industrial and prosthetic design. Examples are to be found in [16], [17], [18], and [19]. The last two give average normal ranges, which an average individual could be expected to satisfy. A dancer would have less restrictive constraints. Reference [16] gives the 2.5 percentile, 50 percentile and 97.5 percentile ranges for men, for women, and for children of four different ages. This reference also has a good bibliography of anthropometric studies. Reference [17] also gives constraint data for different percentiles of the population studied.

III.2-2 Natural Linkages

It will be recalled that the prototype model has a coordinate system associated with each joint. When the figure is at rest, the various rotation angles are zero.

However, this resting position, or position of no rotation, may orient the local coordinate system at an angle with respect to the system with which it is linked. The natural orientation or at-rest rotation, which is assumed to be zero degrees in the prototype, will need to be determined and entered as data in an improved program. This writer was unable to find such data in the literature.

When the dancer representation has reached a level of sophistication such that the above-mentioned data is being incorporated into the program, it would be appropriate to experiment with permutations of the order of multiplication of the three matrices that make up any individual rotation matrix. This order remains fixed in the prototype as the rotation about the X axis first, followed by the rotations about Y and the Z. However, it might be discovered, for example, that the movement of the wrist is most naturally described by rotating first about the Z axis, followed by X and then Y. (There are six possible permutations of the three rotations.) Note that, matrix multiplication not being commutative, the product or result will vary, depending on the order in which the rotations take place.

III.2-3 Body Dimensions

Data for such things as limb length, shoulder width, etc., may also be found in anthropometric literature [16], [17]. (Limb length is not truly fixed as an arm or leg moves, but again, it may be assumed constant, probably

without much loss of verisimilitude, particularly in the context of currently available display resolution.)

Using these kinds of measurements, a mathematical representation of an improved stick figure could be created. Or, going one step further, a "fleshed-out" figure could be represented, still using only straight lines.

It is possible within the general framework of the program to create a wire-frame or a polyhedral representation of the dancer. Either is envisioned as a set of rigid, non-intersecting three-dimensional structures. The single vectors of the prototype, each of which is associated with a different coordinate system, would each in the expanded version be a set of vectors, all fixed in one coordinate system. As soon as the algorithm for determining point coordinates has found the primary point coordinate vector, all the other point coordinate vectors fixed in that same coordinate system should be determined. (See Figure II.3-2.) The display algorithm would likewise have to be altered by including for each coordinate system a table of the mapping of the points associated with that system, which mapping is, in the current program, a single vector. A wire-frame or a polyhedral figure could be described, depending on the mapping used. Included in this mapping could be inter-limb connections to create knees, elbows, etc. The vectors or surfaces that comprise these connections are, in effect, flexible coverings of skin.

The curtains and scenery (stage props) description

could be included within this expanded data structure. The various elements of stage and scenery display would be vectors fixed in the stage coordinate system, corresponding to the set of vectors belonging to a single limb.

III.3 Display Considerations

Along with wire-frame representation of the dancer, hidden-line removal would be desirable, so that the dancing figure would be displayed not as a transparent wire-frame structure but rather as a solid structure. Alternatively, if the object points were mapped in a polyhedral representation, hidden surface removal would be appropriate. Either of these approaches, as well as shaded renderings, is possible with existing algorithms.

Another feature that could be added to the prototype program is a provision for windowing. Because of the wrap-around characteristic of the available display hardware, the model will, when moved off the display, reappear on the opposite side. Thus there is a need to use some existing hardware or software which will perform windowing or clipping; that is, that will eliminate portions of the picture outside of the viewing window [11].

III.4 Library-based Program

Ultimately, the type of program under consideration would have to be based on a library of standard movements if it is to be a useful program. The user would select the

desired standard moves, perhaps modify them in some way, and then join them. The user would also have the option of creating new moves, of course, perhaps adding them to his library. Maximum flexibility would have to be allowed by providing options for timing, splicing, repeating, saving, altering and interpolation between movements by the program.

There should be provided a choice of different dancer models, perhaps allowing the choreographer to specify individual body parameters. Different standard scenery elements or the option of uniquely specifying them should likewise be available.

The standard library movements could be created in a number of ways, and it remains for future workers to determine the best approach. Possibilities include the curve-drawing approach of this paper, the anthropometric harness [6], and the joystick. The track ball is another input device which might be considered, although the author found it unsatisfactory in a preliminary trial. Perhaps analysis of multiple photographs of live, moving dancers will provide the best approach to compiling a library of movements.

The method or methods of joining the standard movements might be different from the methods used in creating them. The possibilities include, in addition to those mentioned in the previous paragraph, automatic filling-in and joining by the program and mathematical specifications of the user. This problem of joining or splicing was not studied per se

in this work, but the curve-drawing approach provides a tool for accomplishing this.

The implementation of a graphics program that would be truly useful to choreographers is a profound undertaking that has not yet been achieved. However, computer applications have been thus far limited more by the imagination of men than by the computer hardware, and the realization of a useful choreography program of the type outlined may be realistically anticipated.

BIBLIOGRAPHY

1. Hutchinson, Ann. Labanotation. New York: James Laughlin, 1961.

2. LeVasseur, Paul. "Computer Dance - The Role of the Computer." Impulse - International Exchange in Dance, 1965, p. 25.

3. Beaman, Jeanne. "Implications of the Dance." Impulse - International Exchange in Dance, 1965, p. 27.

4. Noll, A. Michael. "Choreography and Computers." Dance Magazine, January, 1967, p. 43.

5. Noll, A. Michael. "The Digital Computer as a Creative Medium." IEEE Spectrum, October, 1967.

6. Honey, Francis J. "Computer Animation - A New Look." Proc. 7th Meeting UAIDE, 1968.

7. Roberts, Lawrence G. "Homogeneous Matrix Representation and Manipulation of N-dimensional Constructs." Preprint MS-1405, Lincoln Laboratory, M.I.T., Lexington, Massachusetts, May, 1965.

8. Romney, Gordon W. "Computer Assisted Assembly and Rendering of Solids." Ph.D. Thesis, University of Utah Department of Electrical Engineering, August, 1969.

9. Amir-Moez, A. R. and Fass, A. L. Elements of Linear Spaces. New York: Permagon Press, Macmillan Co., 1962.

10. Johnson, Timothy E. "Sketchpad III - A Computer Program for Drawing in Three Dimensions." AFIPS Proc. SJCC, XXIII (1963), 347.

11. Sproull, Robert F. and Sutherland, Ivan E. "A Clipping Divider." AFIPS Proc. SJCC, XXXIII (1968), 765.

12. Spiegel, Murray R. Theory and Problems of Statistics. New York: Schaum Publishing Co., 1961.

13. Weeg, Gerard P. and Reed, Georgia B. Introduction to Numerical Analysis. Waltham, Mass.: Blaisdell Publishing Co., 1966.

14. Copeland, Lee. "Dance, A Graphics Program." Computer Science, University of Utah, Salt Lake City, Utah, 1968.

15. Mahl, Robert. "The Adventures of a Rotation Matrix in the Computer World." Computer Science, University of Utah, Salt Lake City, Utah, Spring, 1969.

16. Dreyfuss, Henry. The Measure of Man. New York: Whitney Library of Design, 1960.

17. Damon, Albert; Stoudt, Howard W.; and McFarland, Ross A. The Human Body in Equipment Design. Cambridge, Mass.: Harvard University Press, 1966.

18. Committee of the California Medical Association and Industrial Accident Commission of the State of California for Standardization of Joint Measurements in Industrial Injury Cases. Evaluation of Industrial Disability. New York: Oxford University Press, 1960.

19. Moore, Margaret Lee. "Measurement of Joint Motion." Physical Therapy Review, XXIX (1949), 195-205, 256-264.

20. Copeland, Lee and Carr, C. Stephen. GS - Graphics System. Technical Report 4-1, Computer Science, University of Utah, November 15, 1967.

21. Reed, Alan C.; Dallin, D. E.; and Bennion, Scott T. A Fortran V Interactive Graphical System. Technical Report 4-4, Computer Science, University of Utah, April 3, 1968.

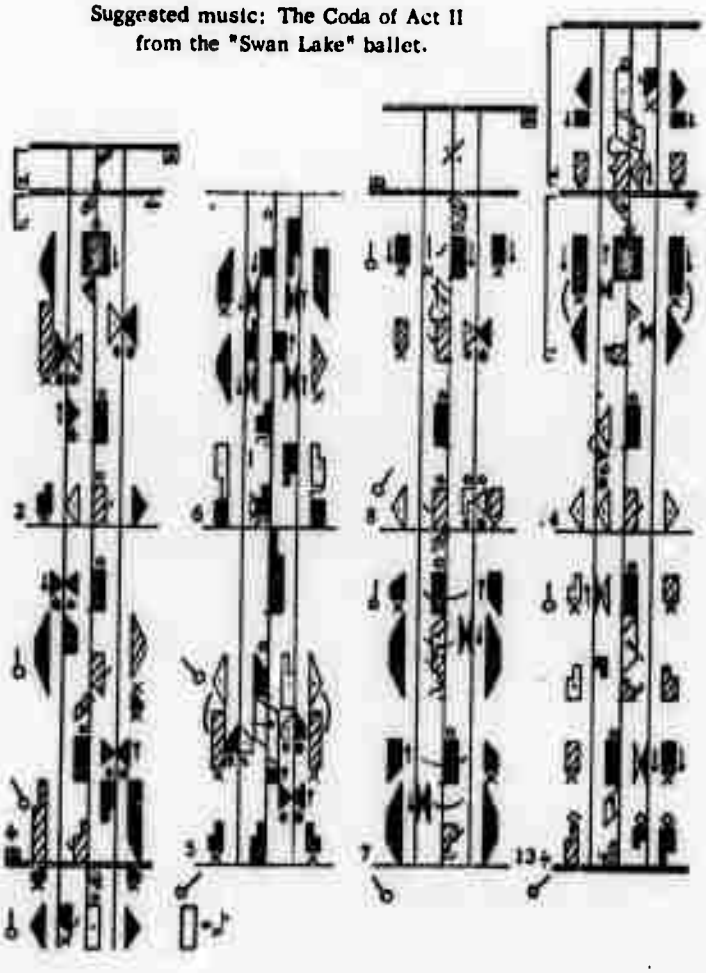
APPENDIX A

An example of Labanotation is shown. This is taken from page 143 of reference [1]. It describes one dancer's movements for several seconds.

This is reprinted with the permission of the publishers, Theatre Arts Books, New York, copyright 1954 by The Dance Notation Bureau, Inc.

STUDY IN BALLET STYLE
(Use of Parts of the Leg)

Suggested music: The Coda of Act II
from the "Swan Lake" ballet.



APPENDIX B

INTERACTIVE GRAPHICS SYSTEM

The graphics system used for this work consists of a cathode ray tube display and several interactive devices which are interfaced to a Univac 1108 computer with a small PDP-8 computer. The graphics system functions under a swap configuration in the 1108, time-sharing not being implemented for the University of Utah's 1108 at this writing.

The display device is an IDI Graphics Terminal with vector line, dashed line, dot and character-generation capabilities. The interactive devices that have been used in this work are a Sylvania Tablet and stylus and a Teletype. Other equipment in the graphics laboratory includes a Univac 1004 card processor and an 1108 console monitor.

Software used is described in [20] and [21]. These sources describe a general purpose set of Fortran-callable subroutines for producing graphics. Also outlined are procedures for interacting by means of the Teletype and Sylvania Tablet. Fortran V is available under the 1108 executive, the above software packages being subsets of Fortran V.

APPENDIX C

PROGRAM LISTING

```
@ ASG D=SUUCC$
@ ASG G=$$SD2$
@ XQT CUR
      IN D
@I FOR FLEVEN
      LOGICAL DEBUG
      COMMON DEBUG.
      DEBUG=.FALSE.
      CALL SYSTEM
      CALL SHOW
      CALL PI
      END

@N FOR SYSTEM
      SUBROUTINE SYSTEM
      CALL RELOAD
      CALL INOUTM
      CALL SETSWP(1,1)
      CALL SETSWP(4,400)
      CALL SETBUF(4,400)
      CALL TABABL
      CALL TABTOL(4)
      RETURN
      END

@N FOR PI
      SUBROUTINE PI
      INTEGER PAGE1(1800)
      INTEGER X,Y,Z
      NAMELIST/N1/ABORT1
      CALL SETLST
      READ(5,N1)
      CALL JUMPS('N1',880)
      IF(FLAG.GT.0.) GO TO 40
      FLAG=1.
      CALL SETDF(PAGE1,INDEX)
      CALL RTURN(1+0,0+0,1+0)
      CALL FLFINS
      CALL LCHAR
      CALL WRITAT(351,880,'A DYNAMIC BODY MODEL ')
      CALL WRITAT(375,752,'TABLE OF CONTENTS ')
      CALL SCHAR
```

```
      CALL WRITAT(351,640,'RIGHT HIP ANGLES ')
      CALL WRITAT(655,640,'2 ')
      CALL WRITAT(655,550,'3 ')
      CALL WRITAT(351,460,'MODEL ')
      CALL WRITAT(655,460,'4 ')
      ISAVE = INDEX
      GO TO 50
   40 INDEX = ISAVE
      CALL RSETDF(PAGE1,INDEX)
   50 CALL SENDF
      CALL CHRINT(1,$70)                    @ TO ABORT
      CALL TABINT(1,$90)
   60 CALL IDLE
      CALL SWAP
      GO TO 60
   70 CALL TTY
   75 CALL INTRET
   80 CALL SETDF(PAGE1,INDEX)               @ WIPE OUT PIC
      STOP                                  @ WHEN ABORT TYPED
   90 CALL GETTAB(X,Y,Z)
  100 CALL GETTAB(I,J,K)
      IF(K.NE.-1) GO TO 100
      IF(Y.GT.600) CALL P2
      IF(Y.GT.515) CALL P3
      IF(Y.GT.420) CALL P4
      GO TO 75
      END

@N FOR P2
      SUBROUTINE P2
      INTEGER*4 X,Y,Z,PAGE2(1800)
      NAMELIST/N2/ABORT2,THRU2,I2,IS2
      CALL SETLST
      READ(5,N2)
      CALL JUMPS('N2',$80,$55)
      IF(FLAG.GT.0) GO TO 40
      FLAG=.1
      CALL SETDF(PAGE2,I2)
      CALL COMAND
      CALL BOXES
      CALL ZLINE(2+1)
      CALL STICK
      CALL RTURN(2+0,1+0,1+0)
      CALL SCHAR
      CALL FLEINS
      CALL WRITAT(676,640,'RT HIP ')
      CALL POS(768,818)
      CALL VEC(713,729)                     @ DRAW LEG
      IS2=I2
      GO TO 50
C ENTER HERE AFTER FIRST CALL
   40 I2=IS2
      CALL RSETDF(PAGE2,I2)
   50 CALL SENDF
      CALL CHRINT(1,$70)                    @ TO ABORT
   55 CALL TABINT(1,$90)
   60 CALL IDLE
      CALL SWAP
```

```
      GO TO 40
   70 CALL TTY
   75 CALL INTRET
   80 CALL SETDF(PAGE2,12)                    @ WIPE OUT PIC
      CALL SENDF
      STOP                                    @ WHEN ABORT TYPED
   90 CALL GETTAB(X,Y,Z)
  100 CALL GETTAB(I,J,K)
      IF(K.NE.-1) GO TO 100
      IF(Y.LT.935) GO TO 110                  @ NOT PAGE TURN
      IF(X.LT.512) CALL P1                    @ TURN PAGE
      CALL P3                                 @ TURN PAGE
  110 IF(Y.LT.875) GO TO 130                  @ NOT DRAW CURVES
      I2=IS2
      CALL GETCRV(PAGE2,I2,2+0)               @ DRAW CURVES
      ISHOW = I2
      CALL CHRINT(1,$70)
      CALL TABINT(1,$90)
      CALL INTRET
  130 IF(Y.LT.780) GO TO 140                  @ NOT ERASE
C ERASE
      GO TO 40
  140 IF(Y.LT.728) GO TO 150                  @ NOT SHOW CURVES
C SHOW CURVES
      I2=ISHOW
      CALL RSETDF(PAGE2,I2)
      CALL SENDF
      CALL INTRET
  150 IF(Y.LT.680) GO TO 160
C RE-USE CURVES
      CALL GETPTS
      CALL INTRET
  160 IF(Y.LT.560) GO TO 75                   @ NOT TABLE CONTENTS
C TABLE OF CONTENTS
      CALL P1
      RETURN
      END

@N FOR P3
      SUBROUTINE P3
      INTEGER X,Y,Z,PAGE3(1800)
      COMMON/US/U(12,4,4)
      NAMELIST/N3/ABORT3,PLOT3,I3,IS3,U
      CALL SETLST
      READ(5,N3)
      CALL JUMPS('N3',$80,$200)
      IF(FLAG.GT.0.) GO TO 40
      FLAG = 1.
      CALL SETDF(PAGE3,I3)
      CALL COMAND
      CALL BOXES
      CALL ZLINE(3+1)
      CALL STICK
      CALL RTURN(3+0,1+0,1+0)
      CALL SCHAR
      CALL FLPINS
      CALL ARITAT(576,640,'RT KNEE ')
      CALL POS(768,818)
```

```
      CALL VEC(743,773)
      CALL VEC(755,724)
      IS3 = I3
      GO TO 50
C  ENTER HERE AFTER FIRST CALL
   40 I3 = IS3
      CALL RSETDF(PAGE3,I3)
   50 CALL SENDF
   55 CALL CHRINT(1,$70)
      CALL TABINT(1,$90)
   60 CALL IDLE
      CALL SNAP
      GO TO 40
   70 CALL TTY
   75 CALL INTRET
   80 CALL SETDF(PAGE3,I3)
      CALL SENDF
      STOP
   90 CALL GETTAB(X,Y,Z)
  100 CALL GETTAB(I,J,K)
      IF(K.NE.-1) GO TO 100
      IF(Y.LT.935) GO TO 110
      IF(X.LT.512) CALL P2
      CALL P4
  110 IF(Y.LT.875) GO TO 130
      I3=IS3
      CALL GETCRV(PAGE3,I3,3+0)
      ISHOW = I3
      CALL CHRINT(1,$70)
      CALL TABINT(1,$90)
      CALL INTRET
  130 IF(Y.LT.780) GO TO 140
C ERASE
      GO TO 40
  140 IF(Y.LT.728) GO TO 150
C SHOW CURVES
      I3=ISHOW
      CALL RSETDF(PAGE3,I3)
      CALL SENDF
      CALL INTRET
  150 IF(Y.LT.680) GO TO 160
C RE-USE CURVES
      CALL GETPTS
      CALL INTPET
  160 IF(Y.LT.560) GO TO 75
C TABLE OF CONTENTS
      CALL P1
  200 CALL PLOTDF
      GO TO 55
      END

@N FOR P4
      SUBROUTINE P4
      INTEGER X,Y,Z,PAGE4(1800)
      REAL LEFT
      COMMON/CP4/PAGE4,IS4,I4
      COMMON/PS/P(12,4,145),JJ
      COMMON/MISC/NN,EYESTG,EYETUR
```

```
      COMMON/ANGLE/ANG(12,3,145)
      NAMELIST/N4/APORT4,THRU4,I4,IS4,NN,P,EYESTG,EYETUB,LEFT,U
      CALL SETLST
      READ (5,N4)
      CALL JUMPS('N4', $80, $55)
      IF (FLAG.GT.0.) GO TO 40
      FLAG = 1.
      CALL SETDF(PAGE4,I4)
      CALL PTURN(4+0,1+0,0+0)
      CALL LCHAR
      CALL FLEINS
      CALL WRITAT(55,896,'SHOW MOVE ')
      CALL WRITAT(55,848,'ROTATE X      ')
      CALL WRITAT(55,800,'ROTATE Y      ')
      CALL WRITAT(55,752,'STOP ')
      CALL WRITAT(55,592,'TABLE OF CONTENTS ')
      CALL WRITAT(55,704,'RE-USE ')
      IS4=I4
      GO TO 50
   40 I4=IS4
      CALL RSETDF(PAGE4,I4)
   50 CALL SENDF
      CALL CHRINT(1,$70)
   55 CALL TABINT(1,$90)
   60 CALL IDLE
      CALL SCAP
      GO TO 60
   70 CALL TTY
   75 CALL INTRET
   80 CALL SETDF(PAGE4,I4)
      CALL SENDF
      STOP
   90 CALL GETTAB(X,Y,Z)
  100 CALL GETTAB(I,J,K)
      IF(K.NE.-1) GO TO 100
      IF(Y.LT.935) GO TO 110
      CALL P3
  110 IF(Y.LT.875) GO TO 120
      CALL ACTION
      CALL INTRET
  120 IF(Y.LT.820) GO TO 130
C ROTATE X
C UP=VALUE TYPED IN (IN RADIANS)
      DO 125 J=1,JJ
  125 ANG(1,1,J)=UP
      CALL INTRET
  130 IF(Y.LT.780) GO TO 140
C ROTATE Y
C LEFT=VALUE TYPED (IN RADIANS)
      DO 135 J=1,JJ
  135 ANG(1,2,J)=LEFT
      CALL INTRET
  140 IF(Y.LT.720) GO TO 150
C STOP
      GO TO 60
  150 IF(Y.LT.680) GO TO 160
      CALL GETPTS
      CALL INTRET
```

```
      CALL P1
      RETURN
      END

@N FOR BDATA
      BLOCK DATA
      REAL MN,MX
      INTEGER P
      COMMON/KIDS/KID(12,31
      COMMON/CONS/MN(12,3),MX(12,31
      COMMON/PS/P(12,4,145),JJ
      COMMON/ULS/UL(12,3,3)
      COMMON/MISC/NN,EYESTG,EYETUB
      COMMON/US/U(12,4,4)
C INITIAL POINT COORDINATE VECTORS (LOCAL COORDINATES)
      DATA(((P(J,K,1),K=1,4),J=1,4)/-512,-512,-512,1, 766,700,0,1,
     X 0,-100,0,1,  0,-100 ,0,1/ ,
     X (((P(J,K,11,K=1,4),J=5,121/0,-100,0,1,  0,-100,0,1,  0,140,0,
     X -110,-25,0,1,  110,25,0,1,  0,50,0,1,  -42,60,0,1, 42,60,0,1.
C ANGLE CONSTRAINTS
     X (((MN(J,K),MX(J,K1,K=1,31,J=3,4)/ -2.09440, .26180, -1.309,
     X .26180,-2.6180,.52360, 0., 2.0944, -.26180, .26180  ,
     X -.03, .03/
C NATURAL LINKAGE ROTATION MATRICES
      DATA(((UL(J,K,L),L=1,3),K=1,3),J=1,31/1.,0.,0.,0.,1.,0.,0.,0.
     X 1.,0.,0.,0.,1.,0.,0.,0.,1.,1.,0.,0.,0.,1.,0.,0.,0.,1./
      DATA((UL(J,K,K),K=1,31,J=4,121/1.,1.,1., 1.,1.,1., 1.,1.,1.,
     X,1.,  1.,1.,1., 1.,1.,1., 1.,1.,1., 1.,1.,1., 1.,1.,1./
C TREE STRUCTURE OF MODEL
      DATA((KID(J,K),K=1,31,J=1,121/0,0,2,1,0,3,2,5,4,3,0,0,2,7,6,5
     X 7,0,8,7,9,0,7,10,0,7,0,11,10,12,0,10,0,0/
C MISC. CONSTANTS:  NN=# PTS THINNED TO, NN .LE. 25,
C EYESTG AND EYETUB= ABS. DISTANCE TO STAGE AND DISPLAY TUBE
      DATA NN,EYESTG,EYETUB/25,20.,1.4/U(1,4,4)/1./
      END

@N FOR GETCRV
      SUBROUTINE GETCRV(DF,INDEX,IP)
C THIS READS 3 CURVES IN ANY ORDER AND STORES X'S IN SAVEX(JOINT,80
C AFTER HITTING DRAW-CURVES LIGHT BUTTON,
C ONE OR MORE CURVES MAY BE REDRAWN.
C AFTER RE-DRAWING,HIT KEEP-CURVES LIGHT BUTTON.
      LOGICAL DEBUG
      INTEGER DF(1800),INDEX,X,Y,Z,TEMP
      COMMON DEBUG
      COMMON/CSX/SAVEX(3,1801,SAVEY(3,1801,IS(31
      NAMELIST/NC/ABORT
      CALL SETLST
      READ(5,NC)
      CALL JUMPS('NC',$1001
      CALL CHRINT(1,$90)
      IF(IP.EQ.IPSAVE) GO TO 2
      IPSAVE=IP
      DO 1 K=1,3
    1 IS(K)=0
    2 K=1                                    @ TRY BOX # 1
    5 CALL TABINT(1,$201
```

```
   10 CALL IDLE
      CALL SWAP
      GO TO 10
   20 CALL GETTAB(X,Y,Z)                    @ GET A POINT
      IF(Y.LT.828) GO TO 21
      IF((Y.GT.875).OR.(X.GT.512)) GO TO 21
C LIGHT BUTTON FOR DEEP CURVES
      CALL GETANG(IP+1)
C IF EXPANDING PROGRAM, CHANGE FOLLOWING INTEGER.
      IF(IP.EQ.3) CALL GETPTS
   22 CALL GETTAB(X,Y,Z)
      IF(Z.NE.-1) GO TO 22
      GO TO 50
   21 TEMP=(K-1)*356
      J=0
      IF(X.LT.(64+TEMP).OR.X.GT.(256+TEMP))GO TO 40
      GO TO 28                                @ RIGHT BOX
   27 CALL GETTAB(X,Y,Z)                      @ GET A POINT
   28 IF(Z.EQ.-1) GO TO 30
      IF(Y.LT.80 .OR. Y.GT.462) GO TO 27  @ OUT OF RANGE
      IF(X.LT.(64+TEMP).OR.X.GT.(256+TEMP)) GO TO 27
      J=J+1
      SAVEX(K,J)=X
      SAVEY(K,J)=Y
      GO TO 27
C DISPLAY DOTTED LINE
   30 IS(K)=J
      I=INDEX
      CALL RSETDF(DF,I)
      DO 35 Y=1,3
      J=IS(Y)
      IF(J.EQ.0) GO TO 35
      DO 34 X=1,J
   34 CALL DOT(IFIX(SAVEX(Y,X)),IFIX(SAVEY(Y,X)))
   35 CONTINUE
      CALL SENDF
      CALL INTRET
   40 K=K+1                                   @ TRY ANOTHER BOX
      IF(K.EQ.4) K=1
      CALL INTRET                             @ RE-ENTER AT $20
C ALL THROUGH
   50 INDEX=1
      IF(DEBUG) CALL TYPOUT('GOTCRV  ')
      RETURN
   90 CALL TTY
      CALL INTRET
C EMERGENCY EXIT
  100 STOP
      END

@N FOR GETANG
      SUBROUTINE GETANG(IPAGE)
C FROM ARRAY XN,YN THIS COMPUTES THREE ANGLE ARRAYS
C THIS SUBROUTINE USES ANGLE CONSTRAINTS
      LOGICAL DEBUG
      REAL MN,MX,M,LX
      COMMON DEBUG
      COMMON/PS/P(12,4,145),JJ
```

```
      COMMON/XNS/XN(3,146),YN(3,146)
      COMMON/ANGLE/ANG(12,3,145)
      COMMON/CONS/MN(12,3),MX(12,3)
      CALL INTRP2
      DO 20 K=1,3                                  @ BOX #=AXIS ABOUT WHICH ROTATE
      DO 20 J=1,JJ                                 @ INDIVIDUAL DATUM
      LX=64+(K-1)*356
      RX=256+(K-1)*356
      M=LX+(ABS(MN(IPAGE,K))*192)/(ABS(MN(IPAGE,K))+MX(IPAGE,K))
      IF(XN(K,J).GE.M) GO TO 10
C NEGATIVE ANGLE
      ANG(IPAGE,K,J)=(M-XN(K,J))*MN(IPAGE,K)/(M-LX)
      GO TO 20
C POSITIVE ANGLE
   10 ANG(IPAGE,K,J)=((XN(K,J)-M)*MX(IPAGE,K))/(RX-M)
   20 CONTINUE
      IF(DEBUG) CALL TYPOUT('GOTANG  ')
      RETURN
      END

@N FOR GETPTS
C UPDATES MATRICES AND FINDS ARRAY OF DISPLAY VECTORS,P.
C CALLS MATMUL,VECMAT AND ROT, WHICH CALLS M3X3M.
C DRIVEN BY THE TABLE ENTERED IN BLOCK DATA.
      SUBROUTINE GETPTS
      INTEGER P
      LOGICAL DEBUG
      COMMON DEBUG
      COMMON/PS/P(12,4,145),JJ
      COMMON/US/U(12,4,4)
      COMMON/KIDS/KID(12,3)
      COMMON/MISC/NN,EYESTG,EYETUB
      CALL TYPOUT('(13HEYE TO STAGE= ,F5.0,5H FT.  16HEYE TO DISPLAY =
     X F5.2, 2H  ) ',EYESTG,EYETUB)
      ET=EYETUB*1000.
      P(1,3,1)=-EYESTG*70.
C TABLE-DRIVEN POINT DETERMINATION
      DO 30 J=2,JJ                                 @ NUMBER OF 'FRAMES'
      NODE=1
C STORE ROTATION PART OF MATRIX IN UPDATE MATRIX
   18 CALL ROT(NODE,J)
      NDAD=KID(NODE,1)
      IF(NODE.EQ.1) GO TO 22
C UPDATE ROTATION PORTION OF MATRIX
      CALL MATMUL(NODE,NDAD)
C FIX REMAINDER OF MATRIX
      DO 20 L=1,4
      U(NODE,L,4)=0.
   20 U(NODE,4,L)=P(NDAD,L,J)
C GET POINT
   22 CALL VECMAT(NODE,J)
C HERE COULD BE DETERMINED COORDINATES OF ALL STRUCTURES
C THAT ARE FIXED IN THIS COORDINATE SYSTEM.
C TEST FOR SON
      IF(KID(NODE,3).EQ.0) GO TO 26
      NODE=KID(NODE,3)
      GO TO 18
C TEST FOR BROTHER
```

```
   26 IF(KID(NODE,2).EQ.0) GO TO 28
      NODE=KID(NODE,2)
      GO TO 26
C TEST FOR FATHER
   28 IF(KID(NODE,1).EQ.0) GO TO 30
C BACK UP
      NODE=KID(NODE,1)
      GO TO 26
   30 CONTINUE
C PERSPECTIVE TRANSFORMATION
      DO 32 J=2,JJ
      DO 32 IP=2,12
      DO 31 L=1,2
   31 P(IP,L,J)=FLOAT(P(IP,L,J))*ET/FLOAT(-P(IP,3,J))+512.
   32 CONTINUE
      IF(DEBUG) CALL TYPOUT('GOTPTS  ')
      RETURN
      END

@N FOR MATMUL
      SUBROUTINE MATMUL(NB,NA)
C UPDATES ROTATION PORTION OF MATRIX
C MULTIPLIES MATRIX U(NB) X U(NA) AND STORES IN U(NB)
      DIMENSION A(3,3)
      COMMON/US/U(12,4,4)
      DO 1 J=1,3
      DO 1 K=1,3
    1 A(J,K)=0.
      DO 2 J=1,3
      DO 2 K=1,3
      DO 2 L=1,3
    2 A(J,K)=A(J,K) + U(NB,J,L)*U(NA,L,K)
      DO 3 J=1,3
      DO 3 K=1,3
    3 U(NB,J,K)=A(J,K)
      RETURN
      END

@N FOR VECMAT
      SUBROUTINE VECMAT(M,I)
C MULTIPLIES INITIAL ROW VECTOR X MATRIX M AND STORES IN VECTOR P(
      INTEGER P
      COMMON/US/U(12,4,4)
      COMMON/PS/P(12,4,145),JJ
      DIMENSION V(4)
      DO 1 L=1,4
    1 V(L)=0.
      DO 3 K=1,4
      DO 2 L=1,4
    2 V(K)=V(K)+FLOAT(P(M,L,I))*U(M,L,K)
    3 P(M,K,I)=IFIX(V(K)+.0005)
      RETURN
      END

@N FOR ROT
      SUBROUTINE ROT(JOINT,J)
C SETS UP AND MULTIPLIES ROTATION MATRICES.
C PLACES PRODUCT IN UPDATE MATRIX.
```

```
      LOGICAL DEBUG
      COMMON DEBUG
      DIMENSION D(3,3), E(3,3), F(3,3)
      COMMON/ULS/UL(12,3,3)
      COMMON/US/U(12,4,4)
      COMMON/ANGLE/ANG(12,3,145)
C SET UP ROTATION MATRIX FOR NATURAL LINKAGE SYSTEM.
      DO 1 L=1,3
      DO 1 K=1,3
    1 F(L,K)=UL(JOINT,L,K)
C SET UP X ROTATION
    2 D(1,1)=1.
      D(1,2)=0.
      D(1,3)=0.
      D(2,1)=0.
      D(2,2)=COS(ANG(JOINT,1,J))
      D(2,3)=SIN(ANG(JOINT,1,J))
      D(3,1)=0.
      D(3,2)=-D(2,3)
      D(3,3)=D(2,2)
C MULTIPLY
      CALL M3X3M(F,D)
C SET UP Y ROTATION
      E(1,1) = COS(ANG(JOINT,2,J))
      E(1,2)=0.
      E(1,3)=-SIN(ANG(JOINT,2,J))
      F(2,1)=0.
      E(2,2)=1.
      E(2,3)=0.
      F(3,1)=-E(1,3)
      E(3,2)=0.
      F(3,3)=E(1,1)
C MULTIPLY
      CALL M3X3M(D,F)
C SET UP Z ROTATION
      D(1,1)= COS(ANG(JOINT,3,J))
      D(1,2)= SIN(ANG(JOINT,3,J))
      D(1,3)= 0.
      D(2,1)= -F(1,2)
      D(2,2)=F(1,1)
      D(2,3)=0.
      D(3,1)=0.
      D(3,2)=0.
      D(3,3)=1.
C MULTIPLY ROTATION MATRICES
      CALL M3X3M(E,D)
C STORE IN UPDATE MATRIX
      DO 10 L=1,3
      DO 10 K=1,3
   10 U(JOINT,K,L) = D(K,L)
   12 RETURN
      END

@N FOR M3X3M
      SUBROUTINE M3X3M(B,A)
C MULTIPLIES MATRIX B X A AND STORES IN A
      DIMENSION A(3,3),B(3,3),C(3,3)
      DO 1 J=1,3
```

```
      DO 1 K=1,3
    1 C(J,K)=0.
      DO 2 J=1,3
      DO 2 K=1,3
      DO 2 L=1,3
    2 C(J,K)=C(J,K)+B(J,L)*A(L,K)
      DO 3 J=1,3
      DO 3 K=1,3
    3 A(J,K)=C(J,K)
      RETURN
      END

@N FOR ACTION
      SUBROUTINE ACTION
C DISPLAYS MOTION
C DRIVEN BY THE TABLE ENTERED IN BLOCK DATA.
      INTEGER PAGE4(1800),P
      COMMON/CP4/PAGE4,IS4,I4
      COMMON/PS/P(12,4,145),JJ
      COMMON/KIDS/KID(12,3)
      CALL RSETDF(PAGE4,I4)
      DO 3 J=2,JJ
      I4=IS4
      NF = 2
      CALL POS(P(NF,1,J),P(NF,2,J))
   10 NS=KID(NF,3)
      IF(NS.EQ.0) GO TO 15
   12 CALL VEC(P(NS,1,J),P(NS,2,J))
      NF=NS
      GO TO 10
   15 NS=KID(NF,2)
      IF(NS.EQ.0) GO TO 20
      NF=KID(NS,1)
      CALL POS(P(NF,1,J),P(NF,2,J))
      GO TO 12
   20 IF(KID(NF,1).EQ.0) GO TO 25
      NF=KID(NF,1)
      GO TO 15
   25 CONTINUE
      CALL VEC(P(11,1,J),P(11,2,J))
    3 CALL SENDF
      RETURN
      END

@N FOR BOXES
      SUBROUTINE BOXES
      CALL FLEINS
      DO 2   K=1,3
      J=(K-1)*356
      CALL POS(64+J,80)                    @ BOXES
      CALL VEC(256+J,80)
      CALL VEC(256+J,462)
      CALL VEC(64+J,462)
      CALL VEC(64+J,80)
      CALL POS(55+J,175)                   @ ARROWS
      CALL VEC(55+J,330)
      CALL VEC(147+J,318)
      CALL POS(55+J,330)
```

```
      CALL VEC(63+J,318)
      CALL SCHAR
      CALL WRITAT(64+J,60,'MIN ')        @ LEGEND
    2 CALL WRITAT(232+J,60,'MAX ')
      CALL LCHAR
      CALL WRITAT(172,470,'X ')
      CALL WRITAT(528,470,'Y ')
      CALL WRITAT(884,470,'Z ')
      RETURN
      END

@N FOR STICK
      SUBROUTINE STICK
C FIGURE
      CALL FLEINS
      CALL POS(768,901)
      CALL VEC(798,930)
      CALL VEC(798,930)
      CALL VEC(768,901)
      CALL DOT(761,923)
      CALL DOT(775,923)
      CALL POS(765,914)
      CALL VEC(771,914)
      CALL POS(768,901)
      CALL VEC(768,704)
C AXES
      CALL POS(828,576)                  @ X
      CALL VEC(768,576)
      CALL VEC(768,436)                  @ Y
      CALL POS(748,576)
      CALL VEC(750,544)                     @ Z
      CALL SCHAR
      CALL WRITAT(838,572,'X ')
      CALL WRITAT(744,641,'Y ')
      CALL WRITAT(736,519,'Z ')
      RETURN
      END

@N FOR ZLINE
C ENTERS DASHED LINE OF DISPLAY BOXES IN DISPLAY FILE.
      SUBROUTINE ZLINE(K)
      CALL WRITAT(351,550,'RIGHT KNEE ANGLES ')
      REAL MN,MX
      COMMON/CONS/MN(12,3),MX(12,3)
      CALL FLEINS
      X=ABS(MN(K,1))/(ABS(MN(K,1))+MX(K,1))
      Y=ABS(MN(K,2))/(ABS(MN(K,2))+MX(K,2))
      Z=ABS(MN(K,3))/(ABS(MN(K,3))+MX(K,3))
      J=X*192+64
      CALL POS(J,80)
      CALL DASH(J,462)
      J=Y*192+64+356
      CALL POS(J,80)
      CALL DASH(J,462)
      J=Z*192 +64+2*356
      CALL POS(J,80)
      CALL DASH(J,462)
      RETURN
```

```
      END

@N FOR COMAND
      SUBROUTINE CO*AND
C ENTERS COMMAND QUADRANT IN DISPLAY FILE
      CALL LCHAR
      CALL FLEINS
      CALL *RITAT(54,496,*DRA@ CURVES *)
      CALL WRITAT(55,440,*KEEP CURVES *)
      CALL CRITAT(55,800,*ERASE *)
      CALL *RITAT(55,252,*SHO@ *)
      CALL WRITAT(55,704,*RE-USE *)
      CALL *RITAT(55,592,*TABLE OF CONTENTS *)
      RETURN
      END

@N FOR PTURN
      SUBROUTINE PTURN(NUMP,LEFTA,RIGHTA)
C ENTERS PAGE NUMBER AND TURN IN DISPLAY FILE
      INTEGER RIGHTA
      IF(LEFTA.EQ.0) GO TO 2
      CALL FLEINS
      CALL POS(488,966)                    @ LEFT ARROW
      CALL VEC(472,966)
      CALL VEC(477,970)
      CALL POS(472,966)
      CALL VEC(477,962)
    2 IF(RIGHTA.EQ.0) GO TO 3
      CALL FLEINS
      CALL POS(536,966)                    @ RIGHT ARROW
      CALL VEC(552,966)
      CALL VEC(547,962)
      CALL POS(552,966)
      CALL VEC(547,970)
    3 CALL SCHAR
      CALL FLEINS
      CALL WRITAT(496,961,'TURN ')
      CALL LCHAR
      CALL WRITAT(496,982,'(2HP,,I2,1H )',NUMP  )
      RETURN
      END

@N FOR EVEN
      SUBROUTINE EVEN
C FINDS ARRAY X, EQUIDISTANT BY SCOPE Y'S
C INCOMING ARRAY SAVEX,SAVEY, EACH IS LONG
C OUTGOING ARRAY  X,Y, EACH NN LONG
C SAVEY IS SCALED TO RANGE BETW. 1-NN,
C THEN THINNED TO VALUES CORRESPONDING TO SCALED
C VALUES CLOSEST TO THESE INTEGERS.
      LOGICAL DEBUG
      COMMON DEBUG
      COMMON/CSX/SAVEX(3,180),SAVEY(3,180),IS(3)
      COMMON/XS/X(3,25),Y(3,25)
      COMMON/MISC/NN,EYESTG,EYETUR
      DO 20 NBOX=1,3
      Y(NBOX,1)=SAVEY(  NBOX,1)
      X(NBOX,1)=SAVEX(  NBOX,1)
```

```
      KK=1
      IN=IS(NBOX)
      DO 10 K=2,IN
      YNORM=(SAVEY(  NBOX,K)-80.)*NN/383.  @ RUNS FROM  1-NN, NOT INTE
      IF(YNORM.LT.FLOAT(KK)) GO TO 10
      KK=KK+1
      Y(NBOX,KK)= SAVEY(  NBOX,K)
      X(NBOX,KK)=SAVEX(  NBOX,K)
   10 CONTINUE
   20 CONTINUE
      IF(DEBUG) CALL TYPOUT('EVENED  ')
      RETURN
      END

@N FOR INTRP2
      SUBROUTINE INTRP2
C POINTS EVENLY THINNED, THEN FILLED IN AGAIN.
      LOGICAL DEBUG
      COMMON DEBUG
      COMMON/XNS/XN(3,146),YN(3,146)
      COMMON/XS/X(3,25),Y(3,25)
      COMMON/GAUS/B(3,4),NB,A(4)
      COMMON/LST/N,NBOX
      COMMON/PS/P(12,4,145),JJ
      COMMON/MISC/NN,EYESTG,EYETUB
      IF(DEBUG) CALL TYPOUT('BEGIN INT2  ')
      CALL EVEN
C FIT EACH 4 PTS TO LST SQUARE PARABOLA AND
C INTERPOLATE 5 PTS BETWEEN 2 AND 3.
C FIRST AND LAST INTERVALS ARE TREATED BY LINEAR INTERPOLATION.
      N=4                                  @ NO. PTS USED TO FIT PARABOLA
      NNM = NN-1
      NNM2=NNM-1
C JJ COMPUTED
      JJ=(NN-1)*6+1                        @ *** JJ=NO. FRAMES ***
      DO 30 J=1,3                          @ BOX #
      NB=3
      DO 20 IJ=1,NNM,NNM2                  @ MATCH NODES
      IJP=IJ+1
      DO 20 K=IJ,IJP
      IK=(K-1)*6+1
      YN(J,IK)=Y(J,K)
   20 XN(J,IK)=X(J,K)
      DO 30 K=1,NNM,NNM2
      DO 30 L=2,6                          @ FILL IN 5 PTS
      Y2=Y(J,K+1)
      Y1 = Y(J,K)
      IF(ABS(Y2-Y1).GT.(2.E-13)) GO TO 28
      IF(DEBUG) CALL TYPOUT('Y2-Y1=0  ')
      Y2=Y2+2.E-13
   28 II=(K-1)*6 +L
      YN(J,II)=Y1+(L-1.)*(Y2-Y1)/6.
   30 XN(J,III=(YN(J,II)-Y1)*(X(J,K+1)-X(J,K))/(Y2-Y1)+X(J,K)
      DO 40 NBOX=1,3
      DO 40 K=3,NNM
      CALL LSTSQ(K)
      II=(K-1)*6 +1
      YN(NBOX,II)=Y(NBOX,K)
```

```
      XN(NBOX,II)=X(NBOX,K)
      DO 40 J=1,5
      L=II-6+J
      YN(NBOX,L)=Y(NBOX,K-1)+J*(  Y(NBOX,K)-Y(NBOX,K-1))/6.
      TEMP=YN(NBOX,L)
   40 XN(NBOX,L)=A(1)+A(2)*TEMP+A(3)*TEMP*TEMP
      L=NNM*4    +1
   90 RETURN
      END

@N FOR LSTSQ
      SUBROUTINE LSTSQ(K)
C SCHAUM STATISTICS,SPIEGEL,P.221
C GET CONSTANTS FOR  X=A1 + A2*Y + A3*Y**2
C N = NUMBER OF POINTS USED IN FITTING ABOVE PARABOLA
C X IS DEPENDENT VARIABLE.  K IS 3RD PT. OF PARABOLA.
      LOGICAL DEBUG
      COMMON DEBUG
      COMMON/LST/N,NBOX
      COMMON/GAUS/B(3,4),NB,A(4)
      COMMON/XS/X(3,25),Y(3,25)
C FIND COEFFICIENTS OF NORMAL EQUATIONS
      DO 10 J=1,3
      DO 10 L=1,4
   10 B(J,L) = 0.
      B(1,1)=N
      DO 20 J=1,N
      XP=X(NBOX,K+J-3)
      YP = Y(NBOX,K+J-3)
      B(1,2)=B(1,2)+YP
      B(1,4)=B(1,4)+XP
      B(2,4)=B(2,4)+YP*XP
      YP=YP*YP
      B(1,3)=B(1,3)+YP
      B(3,4)=B(3,4)+YP*XP
      YP=YP*Y(NBOX,K+J-3)
      B(2,3)=B(2,3)+YP
      YP=YP*Y(NBOX,K+J-3)
   20 B(3,3)=B(3,3)+YP
      B(2,2)=B(1,3)
      B(3,1)=B(1,3)
      B(3,2)=B(2,3)
      B(2,1)=B(1,2)
C FIND A'S BY GAUSS ELIMINATION, BACK-SUBSTITUTION.
      CALL GAUSS
      RETURN
      END

@N FOR GAUSS
      SUBROUTINE GAUSS
C WEEG AND REED , P. 100
C SOLVES SYSTEM OF EQUATIONS BY
C GAUSS ELIMINATION AND BACK-SUBSTITUTION.
      LOGICAL DEBUG
      COMMON DEBUG
      COMMON/GAUS/B(3,4),NB,A(4)
      NP=NB+1
      NM = NB-1
```

```
C REDUCE TO TRIANGULAR FORM
      DO 10 J=1,NM
      JP=J+1
      DO 10 K=JP,NB
      R=B(K,J)/B(J,J)
      DO 10 I=JP,NP
   10 B(K,I)=B(K,I)-R*B(J,I)
C SOLVE SYSTEM BY BACK-SUBSTITUTION
      DO 20 K=NP,1,-1
   20 A(K)=0.
      DO 30 K=NB,1,-1
      KP=K+1
      DO 25 J=NP,KP,-1
   25 A(K)=A(K)-B(K,J)*A(J)
   30 A(K)=(B(K,NP)+A(K))/B(K,K)
      RETURN
      END

01 FOR SHOW
      SUBROUTINE SHOW
C THIS DISPLAYS TITLES USED IN MAKING THE MOVIE.
      INTEGER DF(1800)
      NAMELIST/SH/G1,G2,G3,G4,G5,G6,G7,G8,G9,G10,G11
      CALL SETLST
      READ(5,SH)
      CALL JUMPS('SH',$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11)
      CALL SETDF(DF,1)
      CALL CHPINT(1,$30)
      CALL LCHAR
   20 CALL IDLE
      CALL SHAD
      GO TO 20
   30 CALL TTY
      CALL INTRET
    1 I=0
      CALL WRITAT(392,684,'DYNAMIC STUDIES ')
      CALL WRITAT(498,634,'FOR ')
      CALL WRITAT(296,584,'COMPUTER-AIDED CHOREOGRAPHY ')
      CALL WRITAT(408,450,'CAROL WITHROW ')
      CALL WRITAT(272,400,'DEPARTMENT OF COMPUTER SCIENCE ')
      CALL WRITAT(348,350,'UNIVERSITY OF UTAH ')
      CALL WRITAT(408,300,'FEBRUARY  1970 ')
      CALL SENDF
      CALL INTRET
    2 I=0
      CALL WRITAT(440,584,'THE END ')
      CALL SENDF
      CALL INTRET
    3 I=0
      CALL WRITAT(296,452,'THE MOVEMENT IS SHOWN IN 25 FRAMES. ')
      CALL SENDF
      CALL INTRET
    4 I=0
      CALL WRITAT(352,500,'THE SAME MOVEMENT IS ')
      CALL WRITAT(352,452,'SHOWN IN 20 FRAMES. ')
      CALL SENDF
      CALL INTRET
    5 I=0
```

```
      CALL WRITAT(352,500,'THE SAME MOVEMENT IS ')
      CALL WRITAT(352,452,'SHOWN IN 16 FRAMES. ')
      CALL SENDF
      CALL INTRET
    6 I=0
      CALL PIC(I+4,30,DF,I)
      CALL INTRET
    7 I=0
      CALL PIC(I+4,20,DF,I)
      CALL INTRET
    8 I=0
      CALL PIC(3.,30,DF,I)
      CALL INTRET
    9 I=0
      CALL WRITAT(399,584,'CAMERA ROTATED ')
      CALL SENDF
      CALL INTRET
   10 RETURN
   11 I=0
      CALL WRITAT(270,700,'THIS RESEARCH WAS SUPPORTED IN PART ')
      CALL WRITAT(270,650,'BY THE UNIVERSITY OF UTAH COMPUTER ')
      CALL WRITAT(270,600,'SCIENCE DIVISION AND THE ADVANCED ')
      CALL WRIT T(270,550,'RESEARCH PROJECTS AGENCY OF THE ')
      CALL WRIT T(270,500,'DEPARTMENT OF DEFENSE AND WAS ')
      CALL WRIT T(270,450,'MONITORED BY THE ROME AIR DEVELOPMENT ')
      CALL WRITAT(270,900,'CENTER,GAFB,NEW YORK 13440, UNDER ')
      CALL WRITAT(270,350,'CONTRACT AF30(602)-4277. ')
      CALL END
      CALL INTRET
      END

@I FOR PIC
      SUBROUTINE PIC(E,K,DF,I)
C THIS DISPLAYS DIAGRAM USED IN MAKING MOVIE.
      INTEGER I(1000)
      CALL DOT(256,550)
      CALL POS(256,550)
      CALL DASH(768,550)
      CALL POS(256,550)
      CALL DASH(768,924)
      CALL POS(614,550)
      CALL VEC(614,812)
      CALL POS(768,550)
      CALL VEC(768,924)
      CALL LCHAR
      DO 1 M=10,370,12
    1 CALL DOT(296+M,515)
      CALL WRITAT(212,479,'(17HEYE TO DISPLAY = ,F4.1,18H SCREEN DIAME
     XRS )',E)
      DO 2 M=10,524,12
    2 CALL DOT(296+M,386)
      CALL WRITAT(318,350,'(15HEYE TO STAGE =  ,I4,6H FEET )',K)
      CALL SENDF
      RETURN
      END
```

VITA

Mrs. Withrow was born Carol Ann Thiel on November 3, 1930, in Yuma, Arizona. In 1952 she was graduated with distinction from Arizona State University with a B.S. in biology.

She was employed as a bacteriologist at Dugway Proving Ground, Dugway, Utah, from 1952 until 1953. After her marriage to Dean Withrow, she moved to Salt Lake City. Here she was employed from 1954 until 1957 as a laboratory technician in the Department of Pharmacology of the University of Utah College of Medicine. From 1966 until 1969 she was employed by this same department on a part-time basis as a programmer. During 1969-1970 she held a research assistantship in Computer Science.

The Withrows have three children: Linda, Ann and Kent.